

Système d'exploitation et Administration système UNIX

Travaux Pratiques & Dirigés

Version1.2

Mamadou SOW

mamadou@lipn.univ-paris13.fr



IUT Villetaneuse - Univeristé Paris XIII

99 avenue Jean Baptiste Clément 93430 Villetaneuse

Sommaire

PS :

Image CRIT : Distribution Gentoo	login : n° etudiant mot de passe : code INE
Image GTR5 ou 4 : Distribution Mandrake	login : etudiant mot de passe : iutparis13
Image INFO2 : Distribution Debian Etch	login : etudiant mot de passe : iutparis13

Séance 0 de travaux pratiques

L'installation du système exploitation Linux

1 - Comment se procurer les sources ?

kubuntu 6.10

Mandravia 2007

Debian 3.1

2- Quels sont les choix du mode d'installation ?

a- par réseau

b- par CD

3- Quels sont les choix sur les partitions et le type de partitionnement ?

a- /

b- swap

c- et autres

4 – Quels sont les paquets à installer ?

Séance 1 de travaux pratiques

L'utilisation de l'arborescence depuis le shell

Exercice A -- 1 Connectez-vous à un poste.
Une fois connecté(e), lancez la commande qui donne le nom de la machine sur laquelle on travaille.
Que vous renvoit-elle pour votre poste ?

Déconnectez-vous.
Vérifiez le résultat de la commande précédente.

Exercice A -- 2 Connectez-vous.

Une fois connecté(e), ouvrez une fenêtre shell. Quel est le prompt dans cette fenêtre ?

Où vous trouvez-vous dans l'arborescence Unix ?

Quels fichiers trouvez-vous chez vous ? De quels types sont-ils ?
Etes-vous sûr(e) d'avoir vu tous les fichiers ?

Exercice A -- 3 Remontez d'un niveau dans l'arborescence.
Quel est le chemin de cet endroit ? On répondra par un chemin absolu et par un chemin relatif à votre répertoire d'accueil.
Que trouvez-vous à cet endroit ?

En remontant d'un niveau, le répertoire où l'on arrive a pour chemin relatif « ~/.. ». La commande pwd donnera le chemin absolu.

Remontez d'un niveau dans l'arborescence.
Quel est le chemin de cet endroit ? On répondra par un chemin absolu et par un chemin relatif à votre répertoire d'accueil.

Que trouvez-vous à cet endroit ?

En remontant d'un niveau, le répertoire où l'on arrive a pour chemin relatif « ~/../.. ». La commande pwd donnera le chemin absolu.

Remontez d'un niveau dans l'arborescence.
Quel est le chemin de cet endroit ? On répondra par un chemin absolu et par un chemin relatif à votre répertoire d'accueil.
Que trouvez-vous à cet endroit ?

En remontant d'un niveau, le répertoire où l'on arrive a pour chemin relatif « ~/../../.. ». La commande pwd donnera le chemin absolu.

Au cours de cet excerice, on aura noté la façon dont le prompt du shell change au fur et à mesure que l'on se déplace dans l'arborescence Unix :

Exercice A -- 4 Essayez plusieurs méthodes pour revenir dans votre répertoire personnel.
Laquelle est la plus simple ?

Pour revenir chez soi, plusieurs méthodes :

Exercice A -- 5 Donnez le nom de 3 fichiers présents dans le répertoire d'accueil de l'utilisateur "sow".

Exercice A -- 6 Déplacez-vous chez l'utilisateur « sow ».
Pouvez-vous y faire « ls » ?

Déplacez-vous dans le répertoire « ARS/tp1/ ».
Qu'y trouvez-vous ?

Quels sont les droits du répertoire « aa » ?
Que contient-il ?
Pouvez-vous accéder à ce repertoire ?
Que renvoit «ls » dans ce répertoire ?
Que renvoit « ls -l secret » ?
Pouvez-vous lire le contenu du fichier « secret » ?

Quels sont les droits du répertoire « bb » ?

Que contient-il ?
Pouvez-vous accéder à ce repertoire ?
Que renvoit «ls » dans ce répertoire ?
Que renvoit « ls -l secret » ?
Pouvez-vous lire le contenu du fichier « secret » ?

Exercice A -- 7 Placez-vous dans votre répertoire personnel.
Remontez d'un niveau.
En principe, vous avez déjà identifié ce que l'on trouve ici.
Remontez d'un niveau à nouveau.
En principe, vous avez encore aussi déjà identifié ce que l'on trouve ici.

Comparez-vos déductions avec le contenu du fichier « /etc/passwd » ou _etc_passwd dans ~sow/ARS/tp1.
ATTENTION : le fonctionnement des machines Linux de ARS fait que les comptes ne sont pas définis dans « /etc/passwd » ; ils seront affichés si vous faites la commande « ypcat passwd »

Comprenez-vous mieux la façon dont « cd ~utilisateur » fonctionne ?

Exercice A -- 8 Sans les essayer, pensez-vous que les commandes suivantes fonctionnent ?

```
ls ~sow
more ~sow/ARS/tp1/uid.c
cat ~sow/ARS/tp1/uid.c
echo ~sow
cd ~sow/ARS/tp1/
```

Vérifiez votre avis en essayant les commandes.

Que déduisez-vous en ce qui concerne l'écriture « ~sow » ? Est-elle liée d'une quelconque façon aux différentes commandes essayées ?

Exercice A -- 9 Il existe plusieurs shells Unix. Quel est le votre ?
Pour cela faites « echo $SHELL ».

Séance 2 de travaux pratiques

Les commandes de base (1)

Avant propos Créez le répertoire « tp2 » et travaillez dedans afin de ne pas polluer votre homedir avec les fichiers générés par les exercices.

Exercice A -- 1 : Notion de répertoire courant Affichez le nom du répertoire courant avec la commande pwd (print working directory).

Changez votre répertoire courant pour remonter dans le répertoire parent avec la commande cd (change directory) suivie du nom du répertoire.

Changez votre répertoire courant pour revenir à votre répertoire initial à l'aide de la commande cd sans argument. Vérifiez que vous êtes bien revenu à votre point de départ.

Exercice A -- 2 : Création d'une arborescence Placez vous dans votre répertoire d'accueil.

Soit la succession suivante de commandes :

```
cp /etc/hosts a
mkdir b c
cd b
cp ../a d
mkdir ../e f
cd
cp a b/f/g
cd b/f
cp g ../../e
cd ..
rm ../a
rmdir ../c
mv ../e/g ../e/x
```

Dessinez l'arborescence résultante.
Quel est le répertoire courant à la fin de l'opération ?

Il n'est pas nécessaire de taper les commandes pour faire cet exercice.

Exercice A -- 3 : Création d'une arborescence Quelle séquence de commandes vous permet de créer l'arborescence de la figure ci-dessous :

1-	a/	b/a/b/	b/c/	c/d/
2-	a/b	b/a/a	b/c/a	b/c/c

Exercice A -- 4 : Quelques commandes Avec la commande who, obtenez la liste des utilisateurs connectés.

Connectez-vous à une autre machine de la salle de TP (appelons la machine B). Utilisez pour cela la commande ssh.

Regardez alors la liste des utilisateurs connectés sur cette machine B.

Exercice A -- 5 : Apprentissage de l'éditeur Rentrez le texte fourni en annexe (c'est un programme en langage C)dans un fichier nommé bataille.c.

Pour vérifier si vous l'avez bien tapé, compilez-le avec la commande : cc -o bataille bataille.c Si le compilateur ne dit rien, vous pouvez essayer le programme en tapant ./bataille.

Exercice A -- 6 : Substitutions On désire remplacer les constantes de type caractère ' ', B, *, et X, par des valeurs symboliques. Pour cela, on définit les valeurs symboliques en début du programme :

```
#define RIEN         ' '
#define BATEAU       'B'
#define DEJA_JOUE    '*'
#define BATEAU_COULE 'X'
```

Ajoutez ces définitions en tête du programme, puis remplacez toutes les occurrences des valeurs originales par les noms symboliques à l'aide de la commande s de l'éditeur.

Exercice A -- 7 : Répétition d'une commande A l'aide des commandes de recherche (/), de répétition d'une commande (.) et de recherche d'occurrence suivante (n), modifiez toutes les occurrences de la variable i par x.

Séance 3 de travaux pratiques

L'arborescence UNIX et le shell SH

Il est conseillé de lire la totalité des sujets avant de commencer les exercices.

Avant propos Créez le répertoire « tp3 » et travaillez dedans autant que possible afin de ne pas polluer votre homedir avec les fichiers générés par les exercices.

Exercice B -- 1 : sed Soit le fichier contenant les lignes :
asterix ; 20
tintin / haddock ; 20
tif / tondu ; 30
theodore poussin ; 40
spirit ; 30

Avec la commande sed, faites les modifications suivantes (elles seront traitées indépendamment les unes des autres) :

remplacer les lettres « ; » par un « ! »
remplacer les lettres « t » en début de ligne par un « T »
remplacer les lettres « / » précédé d'un espace par une virgule
remplacer les lettres « / » par un « + » et les lettres « ; » par un « @ »
ne pas afficher les lignes 2 à 4

Exercice B -- 2 : sed avancé / répétition de motifs (1) Soit le fichier contenant les lignes suivantes :
A B
C D
E F
G H

Avec la commande sed, faites :
sed -e 's/\(^.\)/\1\1/g' fichier
sed -e 's/\(^.\)/\1\1\1\1/g' fichier
sed -e 's/\(^.\) \(.\)/\2\2 \1\1/g' fichier
Que constatez-vous ?
Comment appele-t'on l'écriture « ^. » ?
Constatez vous un rapport entre les écritures « \(^.\) » et « \1 » ? entre les écritures « \(.\) » et « \2 » ?

Aidez-vous de la page de manuel de sed au besoin.

Exercice B -- 3 : sed avancé / répétition de motifs (2) Reprenez le fichier contenant les lignes :
asterix ; 20
tintin / haddock ; 20
tif / tondu ; 30
theodore poussin ; 40
spirit ; 30

A la lumière de l'exercice B2, avec la commande sed, faites les modifications suivantes (elles seront traitées indépendamment les unes des autres) :

remplacer l'avant dernier chiffre par « 19NN » où N est l'avant dernier chiffre
échanger les deux parties de part et d'autre du point virgule
enchainer les 2 modifications précédentes :

Exercice B -- 4 : sed avancé / répétition de motifs (3) Sauvez le résultat de la commande « ypcat passwd » dans le fichier «passwords» au moyen d'une redirection (sinon avec ../tp3/_etc_passwd). Si cela n'a pas été vu en cours, demander des explications aux enseignants.

Comment désigner chaque champ de « passwords » au moyen de regexp ?
Indication : quel est le caractère obligatoirement absent d'un champ ?
En déduire une regexp décrivant les caractères présents et permettant de manipuler via sed chaque champs.

En partant du fichier « passwords », affichez pour les membres de ARS, l'état civil et le login (dans cet ordre). On utilisera le caractère point-virgule comme séparateur (comme si on voulait exporter le résultat vers Excel). On utiliser un pipe du shell. Si cela n'a pas été vu en cours, demander des explications aux enseignants.

Séance 4 de travaux pratiques

Les commandes de base (2)

Avant propos Créez le répertoire « tp4 » et travaillez dedans autant que possible afin de ne pas polluer votre homedir avec les fichiers générés par les exercices.

On rappelle que « commande1 | commande2 » passe le résultat de la commande 1 à la commande 2, sans entrainer la formation de fichier intermédiaire.

Exercice A -- 1 : pipes Quelles sont, à votre avis, les commandes suivantes qui fonctionnent :

ls -R | more
echo toto | rm
echo toto | mkdir
ls -R | tail -3
cat toto | head -2
ls -Rl | grep rwxr-- | more
grep cheval toto | sort -r

Exercice A -- 2 : sélection de texte On utilisera le fichier « ~sow/ARS/tp4/data » pour cet exercice :

aaabc2;
absdsdc.
aafdsfsdf;

cbccvf45;
ab333c5;
aa34c;

Affichez les lignes contenant la chaine « ds »

Affichez les lignes contenant un chiffre.

Affichez les lignes contenant une lettre a et contenant plus loin une lettre c.

Affichez les lignes contenant une lettre c suivie d'un chiffre.

Affichez les lignes commençant par autre chose qu'un a.

Affichez les lignes se terminant par un point virgule.

Comment générer ces lignes à partir du fichier «data» et de l'application de grep ?
1:aaabc2;
2:absdsdc.
3:aafdsfsdf;
4:
5:cbccvf45;
6:ab333c5;
7:aa34c;

Exercice A -- 3 : redirection, tri, comptage

Lancez la commande « ypcat passwd > passwords » ou récupérer le fichier
~sow/ARS/tp4/passwords
Que fait-elle ?

On utilisera le fichier « passwords » tout au long du TP.

Le format du fichier généré est :
login : mot de passe : UID numérique : GID numérique : gecos : homedir : shell
Le séparateur des champs est le caractère «:». Les champs sont ainsi accolés sans caractères espace.
Le fichier « passwords » que l'on a généré n'est pas trié.

Triez le fichier par ordre croissant de login.

Triez le fichier « passwords » par ordre croissant de UID.

Combien de lignes le fichier passwords contient-il ?

Exercice A -- 4 : sélection de texte

Sachant que les membres de la formation ARS ont pour homedir quelque chose comme « /net/serveur/home/ars/XXXXXXX », récupérez les lignes concernant ARS depuis le fichier «passwords ». Quelle commande employeriez-vous pour cela ?

Combien de personnes y-a-t-il dans ARS ?

Triez le fichier des membres d'ARS par ordre croissant de nom de famille d'état civil sachant que le gecos dans le descriptif ci-dessus correspond à « Prénom Nom ».

Nota bene : au besoin faites « unset LANG » si vous constatez un affichage par ordre alphabétique où lettres minuscules et majuscules sont indifférenciées.

Exercice A -- 5 : awk Il existe une commande UNIX pratique pour travailler sur un fichier avec des colonnes. Il s'agit de la commande awk. Elle sera décrite plus tard dans le cours.

Créez un fichier data contenant les lignes :

```
or      100000
argent   40000
bois        5
```

Essayez les commandes suivantes :

```
awk '{ print $2 }' data
awk '{ print $2, $1 }' data
awk '{ print $1, "/", $2 }' data
```

et essayez de comprendre ces formes basiques de la commande awk.

Modifiez le fichier data pour qu'il contienne maintenant les lignes :

```
or:100000
argent:40000
bois:5
```

Essayez les commandes suivantes :

```
awk '{ print $2 }' data
awk '{ print $1 }' data
awk -F : '{ print $2 }' data
```

et essayez de comprendre ces formes basiques de la commande awk.

Exercice A -- 6 : pipe de commandes

Affichez les 5 premières lignes des comptes ARS, les lignes étant ordonnées par UID.

Affichez la quatorzième ligne des comptes ARS, les lignes étant ordonnées alphabétiquement.

Au moyen de la commande « cut » (entre autres), affichez les noms de login des gens de ARS en triant le résultat par ordre alphabétique.

Au moyen de la commande « cut » (entre autres), affichez les noms de login des gens de ARS en triant le résultat par UID.

Au moyen de la commande « cut » (entre autres), affichez le nom de login et l'état civil des gens de ARS. Le résultat sera trié par ordre alphabétique décroissant des logins.

Au moyen de la commande « cut » (entre autres), essayez d'inverser les champs 1 et 5 dans l'affichage précédent. Y arrivez-vous ?

Pour réussir à inverser le nom de login et l'état civil, utilisez la commande « awk » (cf exercice A5).
Au passage, on séparera l'état civil du login, cette fois-ci, par le caractère point-virgule (comme si on voulait exporter le résultat vers Excel).

Triez le résultat précédent sur le nom de login.
Pour cela, on pipera le résultat précédent dans une commande de tri.

Exercice A -- 7 : suppression de doublons La commande « uniq » supprime dans un fichier plusieurs lignes consécutives strictement identiques.

Créez un fichier « data » et copiez-y les lignes :
La poste est en greve.
La poste est en greve.
La poste est en greve.
Les eboueurs sont en greve.
Les eboueurs sont en greve.
La poste est en greve.
La poste est en greve.
Appliquez la commande uniq à ce fichier et constatez le résultat.

Triez le fichier puis appliquez la commande `uniq` et constatez le résultat. Vous ferez cela avec un fichier temporaire pour commencer puis sans fichier temporaire.

Séance 5 de travaux pratiques /

La pratique du shell SH

Exercice A -- 1 : redirection de stdin et de stdout

Récupérez et compilez le programme «~sow/tp5/simulation.c».
Appelez l'exécutable généré «simulation».

Lancez le programme sous la forme « ./simulation ». Que fait-il à votre avis ? (vous pouvez en lire le code pour répondre bien sûr).
Dans un fichier nommé «réponses», mettez trois mots à raison d'un mot par ligne.
Que se passe-t-il quand vous faites «./simulation < reponses» ? Pourquoi ? Comment ce mécanisme s'appelle-t-il ?

Lancez maintenant «./simulation << EOF» et tapez ensuite un mot par ligne pendant 3 lignes puis tapez le mot «EOF».
Que se passe-t-il ?

Que se passe-t-il à votre avis si vous faites «./simulation > affichage» ?

Que se passe-t-il si vous faites «./simulation < reponses > affichage» ?

Exercice A -- 2 : job control

Récupérez puis compilez le programme «~sow/tp5/hog.c».
(ne pas tenir compte des messages de warning s'il y en avait)

L'exécutable généré s'appelera «hog»
Lancez le sous le nom « ./hog » avec un paramétre entier inférieur à 10. Observez ce que fait le programme.

Relancez le programme maintenant en l'interrompant avant sa terminaison par «Ctrl-C».
Qu'observez-vous ?

Relancez le programme maintenant en lui donnant 200 comme paramètre et interrompez le avant sa terminaison par «Ctrl-Z».
Qu'observez-vous ?
Refaites cette opération une, deux, trois, quatre, etc. fois de plus jusqu'à... ce que l'on ne puisse plus.
Comprenez-vous maintenant la différence fondamentale entre «Ctrl-C» et «Ctrl-Z» ?

Pour se sortir de tous ses programmes qui ont saturé la machine, faites «jobs». Qu'observez-vous ? Tuez tous les jobs qui sont suspendus. Comment procédez-vous ?

Dupliquez le programme compilé précédemment en lui donnant un autre nom. Par exemple «hog2».

Ouvrez 2 fenêtres terminal.
Dans la première fenêtre, lancez «./hog 1» et suspendez-le par «Ctrl-Z».
Dans la deuxième fenêtre, lancez «./hog2 1» et suspendez-le par «Ctrl-Z». Lancez encore un autre «hog2 1» et suspendez-le aussi par «Ctrl-Z».
Faites «jobs» dans chacune des deux fenêtres. Que voyez-vous. Qu'en déduisez-vous sur ce que renvoit «jobs» ?

Dans la fenêtre 1, donnez la commande pour tuer le job suspendu.
Dans la fenêtre 2, donnez la commande pour tuer le job 1 suspendu. Donnez la commande pour remettre en premier plan, le job 2.

Exercice A -- 3 : job control (2) La commande « sleep » sert à attendre pendant un nombre de secondes spécifié. Par exemple, « sleep 5 » attend 5 secondes. Cette commande va servir de base pour ces manipulations car c'est une commande qui permet de simuler l'exécution d'une longue tâche telle qu'une grosse compilation par exemple.

Lancez la commande « sleep 5 ». Que se passe-t-il ?

Lancez la commande « sleep 500 » en arrière-plan. Vérifiez avec « jobs » que votre commande est toujours là.

Lancez la commande « sleep 5 » en arrière-plan. Que se passe-t-il lorsqu'elle se termine ?

Votre commande « sleep 500 » est toujours active. Mettez-la en avant-plan. Suspendez-la. Faites « jobs ». Quel est son état ? Relancez-la en arrière-plan.

Lancez une deuxième commande « sleep 100 » en arrière-plan. Passez la première en avant-plan. Suspendez-la. Suspendez la deuxième. Reprenez l'exécution de la première en avant-plan. Repassez la première en arrière-plan et reprenez la deuxième en arrière-plan.

Faites « ps » pour controler les processus actifs. Quelles sont les différences avec « jobs » ? Comment faire pour obtenir la liste de tous vos processus ?

Exercice A -- 4 : find, sed A l'aide des commandes « find » et « sed », obtenez une liste indentée des répertoires (et seulement des répertoires) de votre arborescence.

Par exemple, si votre arborescence contient les répertoires « a », « b », « c », « a/d », « a/e », « a/d/f » et « b/g », votre commande devra afficher:

```
a
 d
  f
 e
b
 g
c
```

Indication : étudiez le résultat de la commande find. Quelle forme ont les chemins renvoyés ? Quel est le rapport entre leur profondeur dans l'arborescence et l'affichage voulu ? Quelle partie du chemin renvoyé garde-t-on pour l'affichage ? En quoi la commande sed peut-elle s'occuper de ces différents aspects ?

Exercice A -- 5 : quelques aspects de bash

Quel shell utilisez-vous ? Quelle variable d'environnement vous renvoit-elle l'information ?

Que renvoit «echo $PATH» ?
Que renvoit «type cat» ? Voyez-vous le lien entre ce résultat et la commande «cat» quand vous la tapez ?

La commande «ifconfig» attribue lors du démarrage d'une machine UNIX une adresse réseau.
Pour les élèves les plus avancés, dans quel répertoire se trouve cette commande et pourquoi celui-là ?
Pour les autres élèves, sachez que la commande est dans le répertoire /sbin (ce sera vu en cours plus tard).
Vérifiez cela par la commande «ls -l» qui convient.
La commande est-elle accessible depuis votre shell ? Vérifiez votre réponse par «type ifconfig».

Comment feriez-vous pour rajouter le répertoire «/sbin» à votre chemin de recherche de commandes («PATH») ?
Vérifiez que cela a bien fonctionné (en utilisant «type»).
Comment rendre ce réglage permanent ? Vérifier que votre réglage est bien devenu permanent.

Quels sont les noms invalides dans la liste qui suit pour être utilisés comme noms de variable dans un shell script ?

temp
1abc
abc_d_e

temp.doc
abc#1
avc-d
TeMpVaR
a$3

On rappelle que les règles du langage C pour nommer une variable s'appliquent aussi aux variables du shell.

Exercice A -- 6 : liens symboliques

Récupérez le fichier «~sow/tp5/data».
Que contient-il ?

Faites «ln -s data data2».
Que contient «data2» ?
Utilisez la commande «diff» pour voir s'il existe une différence.
En trouvez-vous une ? Cela est-il logique ? Comment s'appelle l'opération «ln -s» ?
Pour vous en convaincre, modifiez le fichier «data» et affichez ensuite le fichier «data2».
Réciproquement, modifiez le fichier «data2» et affichez ensuite «data». Qu'observez-vous ? OK ?

Qu'affiche «ls -l data data2» ?
Qu'affiche «ls -lL data data2» ?
Cherchez à comprendre les 2 affichages.

De quel type est le fichier «data2» ?

Séance 6 de travaux pratiques

Les shell scripts SH (1)

Exercice A -- 1 : PATH Si un script que vous avez écrit, se trouve dans le répertoire courant, comment le désignez-vous par un chemin relatif ?

Pouvez-vous lancer le script sans le désigner par son chemin relatif ? Vérifiez la valeur de votre chemin de commandes. Contient-il le répertoire courant ? En déduire comment on lance un script ou un exécutable qui se trouve dans le répertoire courant.

Exercice A -- 2 : shell script Ecrivez un shell script. Ce script affichera la phrase « Nous sommes en TP. ». Que faut-il faire pour exécuter ce script ?

Exercice A -- 3 : shell script Ecrivez un shell script. Ce shell script fera les actions suivantes (les 3 actions sont indépendantes les unes des autres) :

copier le fichier « /etc/passwd » en « /tmp/tp6.3 »
extraire de la copie les lignes qui contiennent « oo » (lettre « o », pas le chiffre zéro)
trier la copie par ordre décroissant d'UID (l'UID a été défini dans l'exercice 3 du TP 4) et n'en garder que les 5 premières lignes

Exercice A -- 4 : shell script Ecrivez un shell script. Ce shell script utilisera une boucle « for » pour afficher les noms des fichiers (au sens large) du répertoire courant. La variable de la boucle « for » parcourera le résultat d'une regexp simple qui sélectionne tous les noms des fichiers (au sens large) du répertoire courant. (réponse en 5 lignes)

Exercice A -- 5 : shell script Créez des fichiers dont le nom commence par la lettre « a ». Créez des fichiers dont le nom commence par une lettre autre que « a».

Ecrivez un shell script. Ce shell script s'inspire de l'exercice A4 mais il ne doit afficher que les noms de fichiers (au sens large) qui commencent par la lettre « a ». (réponse en 5 lignes)

Exercice A -- 6 : shell script Ecrivez un shell script. Ce shell script s'inspire de l'exercice A4 mais il ne doit afficher que les noms de fichiers (au sens large) qui contiennent la lettre « a » ou la lettre « b ». (réponse en 5 lignes)

Exercice A -- 7 : shell script Par quelle commande UNIX peut-on tester si une chaine de caractères correspond à un nom de répertoire ? (on pourra prendre « /etc » comme nom de répertoire à tester et « /etc/passwd » comme nom de fichier à tester)

Ecrivez un shell script qui utilise cette commande UNIX et un test « if » pour tester si « /etc » est un répertoire ou pas. On affichera quelque chose selon que le test est positif ou négatif.

Exercice A -- 8 : shell script Ecrivez un shell script. Ce shell script s'inspire des exercices A4 et A7. Il s'agit d'afficher uniquement les noms des objets du répertoire courant s'ils correspondent à des répertoires. (réponse propre en 8 lignes)

Exercice A -- 9 : basename Que renvoient les commandes :

« basename toto.txt .txt »
« basename /a/b/c/toto.txt .txt »
« basename /a/b/c/toto.txt »
« dirname /a/b/c/toto.txt »

Ecrivez un shell script. Ce shell script s'inspire de l'exercice A4. Il s'agit d'afficher uniquement les noms des objets du répertoire courant sans leur extension « .txt ». (réponse propre en 5 lignes) Créez au préalable des fichiers avec l'extension « .txt » pour vérifier que votre script fonctionne bien.

Exercice A -- 10 : shell script Ecrivez un shell script. Ce shell script s'inspire des exercices A4 et A9. Il s'agit d'afficher uniquement les noms des objets du répertoire courant avec puis sans leur extension « .txt ». (réponse propre en 6 lignes)

Ecrivez un shell script. Ce shell script s'inspire des exercices A4 et A9. Il s'agit de renommer les objets d'extension « .txt » en un fichier de même nom avec l'extension ".texte". (réponse propre en 5 lignes)

Séance 7 de travaux pratiques

Les shell scripts SH (2)

Exercice A -- 1 : shell script Ecrivez un script qui prend un paramètre. Ce paramètre correspondra à une chaîne de caractères. Le script recherchera cette chaîne dans le fichier "/etc/passwd".

Le shell script devra afficher le code d'exit de la commande grep utilisée. Rappel : le code d'exit d'une commande est récupéré par "$?".

Exercice A -- 2 : shell script Ecrivez un shell script appelé «arguments» qui rapporterait le nombre d'arguments donnés au script avant de les afficher :

% arguments hello world
Il y a 2 arguments.
Les parametres sont hello world

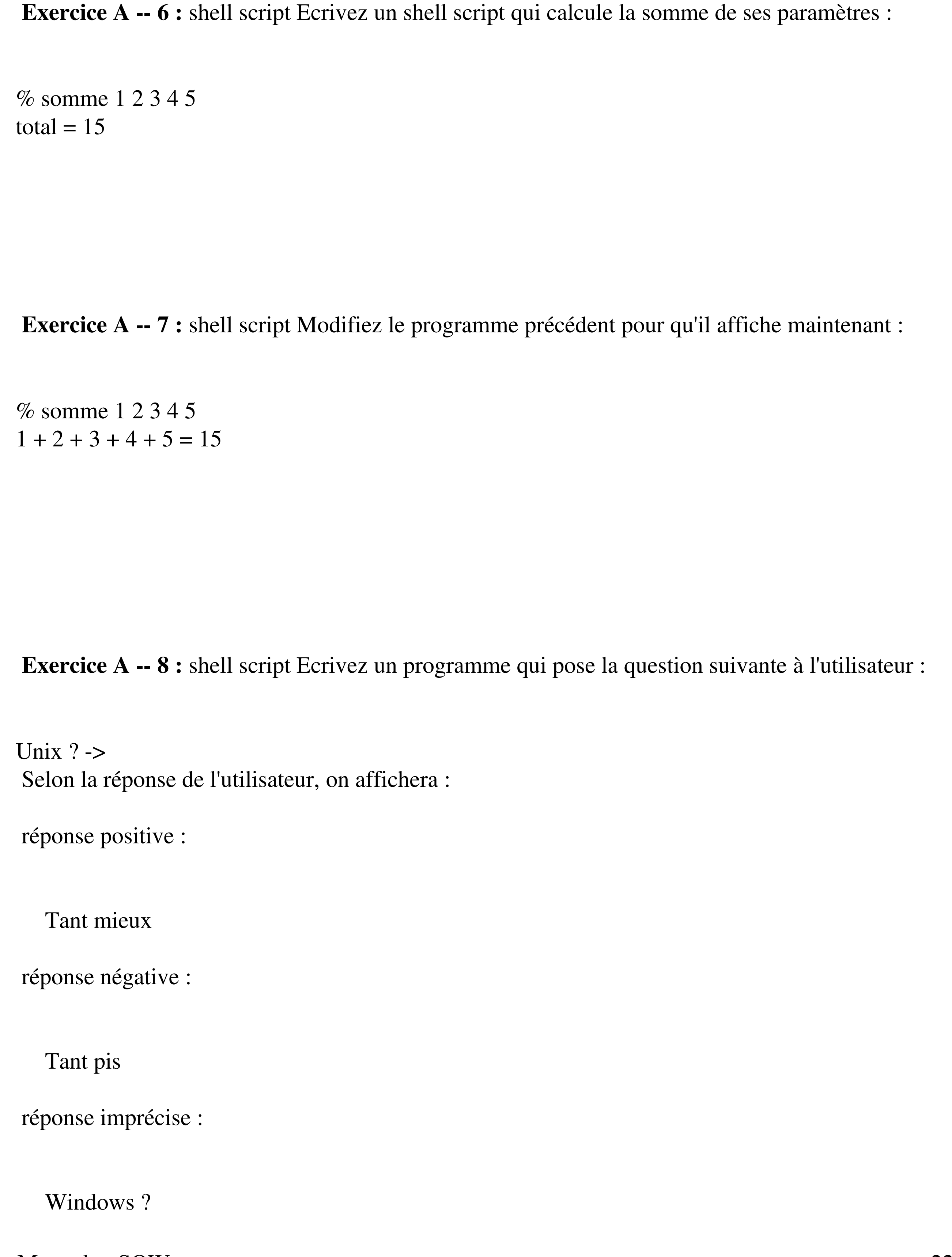
Exercice A -- 3 : shell script Reprenez l'exercice précédent.
Modifiez le de façon à ce que s'il n'y a pas de paramètre, le script affiche maintenant :

Il n'y a pas de parametre.

Exercice A -- 4 : shell script Ecrivez un shell script appelé «existe».
Si un paramètre est passé, on testera si le paramètre correspond à un fichier existant et on affichera une phrase disant si le fichier existe ou pas.

Si aucun paramètre n'est passé, on affichera un message d'erreur indiquant le mode d'emploi du script.

Exercice A -- 5 : shell script Partez du script précédent pour écrire un nouveau script acceptant plus d'un paramètre.



Exercice A -- 6 : shell script Ecrivez un shell script qui calcule la somme de ses paramètres :

% somme 1 2 3 4 5
total = 15

Exercice A -- 7 : shell script Modifiez le programme précédent pour qu'il affiche maintenant :

% somme 1 2 3 4 5
1 + 2 + 3 + 4 + 5 = 15

Exercice A -- 8 : shell script Ecrivez un programme qui pose la question suivante à l'utilisateur :

Unix ? ->
Selon la réponse de l'utilisateur, on affichera :

réponse positive :

Tant mieux

réponse négative :

Tant pis

réponse imprécise :

Windows ?

Exercice A -- 9 : shell script C'est très identique à l'exercice précédent.

Ecrivez un script «somme» qui utilise le fichier «additions» et produit l'affichage comme suit :

% somme additions
2 + 7 = 9
3 + 13 = 16
1 + 28 = 29
...

Pour cela, il faut utiliser une redirection au niveau de la boucle while/do/done. Se reporter au cours pour plus de détails.

Exercice A -- 10 : shell script Compulsez la page de manuel de "test".

Ecrivez un shell script donnant le type (fichier classique, répertoire,lien, etc.) d'un objet fichier au sens large. Vous afficherez un message d'aide si l'on ne donne pas de noms de fichiers. Vous devrez pouvoir tester plusieurs noms de fichiers.

Vérifiez les résultats de votre script sur des fichiers de «/dev» au moyen de la commande «ls». Essayer de découvrir la nature des fichiers testés.

Séance 8 de travaux pratiques

Les commandes UNIX

Il est conseillé de lire la totalité des sujets avant de commencer les exercices.
Avant propos Créez le répertoire « tp8 » et travaillez dedans autant que possible afin de ne pas polluer votre homedir avec les fichiers générés par les exercices.

Exercice A -- 1 : quote, double quote Créez des fichiers appelés « a » et « b». Ils peuvent être vides ou contenir quelques lignes, peu importe.

Créez un fichier appelé « z » et contenant les lignes :

a a a a a a
b b b b b b
$ $ $ $ $ $

Expliquez ce que font les commandes suivantes en vous appuyant sur l'ordre d'évalusation d'une ligne de commande :

« echo '[ab]' »
« echo "[ab]" »
« echo [ab] »
« echo '*' »
« echo "*" »
« echo * »
« sed -e 's/$/#/' z »
« sed -e 's/\$/#/' z »

Exercice A -- 2 : * Travaillez dans un répertoire où il n'y a pas de fichier dont le nom commence par un « a ».

Que font les commandes :

« ls a* »
« echo a* »
« rm -f a* » Pourquoi ?

Créez maintenant un fichier appelé « arbre ».

Que font à nouveau les commandes :

« ls a* »
« echo a* »
« rm -f a* » Pourquoi ?

Exercice A -- 3 : ~ Que font les commandes :

« echo ~sow»
« echo ~foo » Expliquez pourquoi ?

Exercice A -- 4 : shell script exécutable Créez un shell script affichant la date du jour.

Faites ensuite « chmod 644 toto ».

Que se passe-t-il si vous exécutez « ./toto » ? Pourquoi ?

Exercice A -- 5 : interpreteur du shell script Créez un shell script appelé « toto » contenant les lignes :

#!/bin/shell
date
Rendez le script exécutable.
Que se passe-t-il si vous exécutez « ./toto » ? Pourquoi ?

Exercice A -- 6 : pipes Quelle est la différence entre les commandes :

« ls | sort -r »
« sort -r `ls` » Au besoin créez 2 fichiers « A » et « B » contenant chacun

pour « A » :

arbre
ananas

pour « B » :

bretagne
barcelone

Exercice A -- 7 : pipes Quelle est la différence entre les commandes :

« ls | tail -3 »
« tail -3 `ls` » Au besoin créez 5 fichiers « A », « B », « C », « D »e t « E » contenant chacun

pour « A » :

arbre
ananas
argentine
abricot
anglais

pour « B » :

bretagne
barcelone
banane
banque
belote

pour « C » :

cornichon
cerise
coeur
crapaud
cacahuette

pour « D » :

departementale
dossier
dessin
dauphin
dada

pour « E » :

elephant
elastique
elan
emporter
elever

Exercice A -- 8 : tri Soit un fichier contenant des adresses réseau de machines. Pour fixer les idées, ce fichier contient les lignes suivantes

129.199.86.11
129.199.9.11
195.220.117.22
80.2.3.4
134.157.46.129

Triez ce fichier par ordre croissant d'adresses réseau. N'hésitez pas à relire la page de manuel au besoin.

Exercice A -- 9 : boucle for Créez un script qui affiche tous les noms des entrées du répertoire courant à raison d'un nom d'entrée par ligne. On utilisera pour cela une simple boucle for. (solution en 5 lignes)

Exercice A -- 10 : boucle for Créez un répertoire et travaillez dedans. Créez y plusieurs sous répertoires (exemples de noms : « 2001-10 », « 2001-11 », « 2001-12 », « 2002-01 », « 2002-02 », « 2002-03 », etc.) dans lesquels vous créez des fichiers de nom « access.log » et « errors.log ». Vous créérez ces fichiers dans quelques uns de ces répertoires mais pas dans tous, simulant ainsi une répartition plus ou moins aléatoire. Les fichiers créés seront de taille zéro ou contiendront quelques centaines d'octets.

Créez un script qui affiche le nom des fichiers « access.log » uniquement si le fichier a une taille non nulle et dans ce cas là, on compressera le fichier par la commande gzip.

Le script utilisera une simple boucle for. Le script se placera (via la commande cd) dans chaque sous répertoire avant de traiter ou pas le fichier « access.log ».

Séance 9 travaux pratiques

X WINDOW

Exercice B -- 1 : Démarrage d'une session de travail X Quel est le nom du DISPLAY de votre poste ? Pas besoin de se connecter, regardez votre écran.

Il suffit de regarder ce qui est affiché dans la fenêtre d'accueil lorsque vous arrivez devant le poste :

Une fois connecté, une variable du shell redonne le nom du DISPLAY. Laquelle ? De quel type est cette variable ?
La variable d'environnement "DISPLAY" précise le nom du DISPLAY :
% echo $DISPLAY
:0.0
Cette variable est créée par le processus "xdm" ("kdm" sur l'environnement KDE des PC de la salle de TP) et héritée par tous les processus X lancés après.

Votre environnement graphique s'appelle KDE. Au niveau de la fenêtre d'accueil avez-vous une fonctionnalité de type failsafe ?

L'environnement KDE propose une fonctionnalité de type failsafe via la fenêtre d'accueil. Il suffit de choisir l'entrée correspondante dans le menu :

Le mode failsafe est alors indiqué dans la fenêtre :

Lorsque l'on se connecte, on obtient alors juste une fenêtre xterm, sans window manager (voir figure ci-dessous).

Au niveau de la fenêtre d'accueil avez-vous la possibilité de redémarrer la machine ?

L'environnement KDE propose une fonctionnalité de redémarrer la machine grâce au menu Shutdown :

Particularité de Linux : après avoir démarré une session X, que se passe-t-il si vous appuyez sur "Ctrl Alt F1" ou d'une façon plus générale "Ctrl Alt FN avec N variant de 1 à 6 ?
Que se passe-t-il si depuis un écran en mode texte, vous appuyez sur "Ctrl Alt F7" ?

Depuis l'environnement graphique, la séquence de touches « Ctrl Alt FN » avec N variant de 1 à 6 amène sur l'écran texte numéro N.

Depuis l'écran texte numéro N, la séquence de touches « Ctrl Alt FN' » avec N' variant de 1 à 6 amène sur l'autre écran texte numéro N'.

Depuis l'écran texte numéro N, la séquence de touches « Ctrl Alt F7 » ramène sur l'écran de X Window.

Exercice B -- 2 : xterm Lancez l'application xterm en tache de fonds. Comparez rapidement le terminal offert par KDE et xterm.

La fenêtre du terminal KDE a l'aspect suivant :

La fenêtre du terminal xterm a l'aspect suivant :

Les 2 fenêtres sont radicalement différentes dans leur aspect mais offrent cependant une session shell. Les différences sont au niveau des fonctionnalités sous X. Par contre, il n'y a pas de différence au niveau de ce que l'on peut faire dans le shell.

Dans une fenêtre xterm, si l'on appuie sur la touche "Ctrl" en même temps que sur l'un des boutons de la souris, des menus surgissent.

Quels sont les titres de ces menus ?
Jouez avec ces menus.

Bouton de gauche :
L'entrée menu la plus intéressante est "Secure keyboard".

Bouton du milieu :
On notera dans ce menu les entrées "Do Soft Reset", "Do Full Reset", "Reset and Clear Saved Lines".

Bouton de droite :
Ce menu permet de régler les fontes de la fenêtre.

Que font un simple clic, un double clic, un triple clic souris du bouton gauche dans une fenêtre gnome terminal ?

Un simple clic :
il ne se passe rien.

Un double clic :

Un triple clic :

Pratiquez le couper-coller entre deux fenêtres, en variant le contexte du collage (par exemple collez dans un vi en cours, tantôt en mode insertion de texte, tantôt en mode commande).

On sélectionne du texte (couper) en cliquant sur du texte avec le bouton gauche de la souris puis en déplaçant le curseur en maintenant le bouton enfoncé. On relâche le bouton en fin de sélection.

On colle le texte en cliquant avec le bouton du milieu.

Exercice B -- 3 : autorisations de connexion Examinez le contenu de la variable DISPLAY.

Modifiez cette variable (enlevez le :0.0 final, ou changez le nom de votre terminal par n'importe quoi). Que se passe-t-il lorsque vous lancez un programme X (xterm par exemple) ?

D'une façon générale, le DISPLAY peut prendre trois formes :

Première forme : forme concise
% echo $DISPLAY
:0.0

Seconde forme : forme semi-longue
% echo $DISPLAY
ars01:0.0

Troisième forme : forme longue
% echo $DISPLAY
ars01.formation.jussieu.fr:0.0
La forme 1 indique d'utiliser l'écran rattaché à la machine. Sous cette forme, le dialogue entre le client X et le serveur X est optimisé car on sait que l'on n'a pas à utiliser la couche logicielle réseau d'UNIX.

Sous la forme 2 ou 3, le dialogue n'est pas optimisé puisque le client va établir une communication réseau avec le serveur X indiqué dans la variable DISPLAY, même si ce nom correspond au nom de la machine même.

En pratique, sur une machine de la salle de TP, on se voit renvoyer une forme 1 :

% hostname
ars01.formation.jussieu.fr
% echo $DISPLAY
:0.0
Pour modifier le contenu de la variable DISPLAY :
% DISPLAY=ars01
% export $DISPLAY
% echo $DISPLAY
ars01
Lançons un client X, xterm par exemple :
% xterm
xterm Xt error: Can't open display: ars01
Ca ne marche pas car on spécifie un DISPLAY qui a une forme incorrecte et qui ne correspond donc à rien.

Modifiez la variable DISPLAY pour indiquer le terminal de votre voisin (avec son accord bien sûr). Que se passe-t-il lorsque vous lancez un programme X ?

On modifie la variable DISPLAY et on vérifie sa nouvelle valeur par :
% DISPLAY=ars02:0.0
% export $DISPLAY
% echo $DISPLAY
ars02:0.0
Lançons un client X, xterm par exemple :
% xterm
Xlib: connection to "ars02:0.0" refused by server
Xlib: Client is not authorized to connect to Server
xterm Xt error: Can't open display: ars02:0.0
Cela ne marche pas : le serveur X du voisin ne nous autorise pas à se connecter chez lui.

Exercice B -- 4 : autorisations de connexion (2), xhost Avec l'aide de votre voisin, vous allez ouvrir une fenêtre sur son terminal. Demandez-lui de vous ouvrir sa machine via la bonne forme de la commande « xhost ». Affichez un xterm sur son poste. Arrêtez le xterm puis que votre voisin referme son terminal en utilisant la bonne forme de la commande « xhost ».

Admettons que vous êtes sur la machine « ars01 » et que votre voisin est sur la machine « ars02 ».
Votre voisin doit faire :
% xhost + ars01
ars01 being added to access control list
% xhost
access control enabled, only authorized clients can connect
INET:ars01.formation.jussieu.fr

INET:localhost
LOCAL:
Vous faites ensuite sur votre poste ars01 :
% DISPLAY=ars02:0.0
% export $DISPLAY
% echo $DISPLAY
ars02:0.0
% xterm

ce qui donne sur votre écran :

La fenêtre de xterm apparait chez le voisin :

Le voisin arrête le xterm via « exit » dans la fenêtre du xterm.

Le voisin referme l'accès à son serveur X par :
% xhost - ars01
ars01 being removed from access control list
% xhost
access control enabled, only authorized clients can connect
INET:localhost
LOCAL:

Exercice B -- 5 : autorisation de connexion (3), MAGIC COOKIE Cet exercice va maintenant essayer d'accèder au DISPLAY de votre voisin sans passer par l'utilisation de la commande « xauth » mais en utilisant les « MAGIC COOKIES ».

Avec l'aide de votre voisin, vous allez ouvrir une fenêtre sur son terminal.

Il faut d'abord que votre voisin vous communique le cookie de sa session X. Pour cela, il doit utiliser une commande proche de celle donnée en cours qui était « xauth extract - $DISPLAY ». Le résultat en sera redirigé dans un fichier chez lui dont il vous communiquera le nom.

Cela dit, cette commande ne marchera pas du premier coup dans la mesure où le DISPLAY dans l'environnement KDE de la salle de TP ARS est positionné par défaut à la valeur :0.0. Votre voisin devra remplacer l'écriture « $DISPLAY » sous une forme du type « ars01:0.0 » (avec le bon numéro à la place de 01).

Enregistrez maintenant sa clef MAGIC COOKIE dans votre propre fichier de clefs MAGIC COOKIE au moyen de « xauth ».

Que se passe-t-il lorsque vous lancez un programme X ? Demandez à votre voisin de quitter sa session X et d'en ouvrir une nouvelle toujours sur le même poste. En conservant le même cookie, pouvez-vous toujours ouvrir une fenêtre sur son poste ? Retirez le cookie de votre voisin de votre fichier de cookies.

En suivant le commentaire de l'énoncé, votre voisin sur ars02 copie son cookie dans un fichier temporaire nommé cookie chez lui par :

% DISPLAY=ars02:0.0
% export DISPLAY
% xauth extract - $DISPLAY > $HOME/cookie
ou via :

% xauth extract - ars02:0.0 > ~/cookie

Vous enregistrez ensuite son cookie dans votre fichier de cookies par :

% xauth merge - < ~voisin/cookie

Vous réglez alors votre variable DISPLAY à la valeur de celle du voisin par :

% DISPLAY=ars02:0.0
% export $DISPLAY
% echo $DISPLAY
ars02:0.0
% xterm
et votre écran devient :

alors que sur celui du voisin apparait la fenêtre de xterm sur son DISPLAY :

Votre voisin quitte sa session X (il se déconnecte totalement). Il ouvre juste après une nouvelle session X. Un cookie flambant neuf est généré et installé chez lui. Pour s'en convaincre, il suffit au voisin d'extraire son cookie et de comparer au précédent cookie :
% DISPLAY=ars02:0.0
% export DISPLAY
% xauth extract - $DISPLAY > ~/cookie2
% cmp ~/cookie ~/cookie2
des différences sont affichées

Le cookie copié chez vous n'est donc plus désormais celui qui contrôle l'accès à son serveur X.

Pour s'en convaicre, il suffit de vouloir ouvrir un nouveau xterm sur son écran :
% echo $DISPLAY
ars02:0.0
% xterm
Xlib: connection to "ars02:0.0" refused by server
Xlib: Client is not authorized to connect to Server
xterm Xt error: Can't open display: ars02:0.0

Pour retirer le cookie qui ne sert plus à rien, vous faites :
% xauth remove $DISPLAY
XXXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXXX

Vérifiez les droits d'accès de votre fichier de clefs. Vérifiez les droits d'accès du fichier de clefs de votre voisin.

Chez vous puis chez votre voisin :
% ls -l ~/.Xauthority
-rw------- 1 sow1000 1272 Nov 14 10:27 /net/serveur/home/ars/sow/.Xauthority
% ls -l ~voisin/.Xauthority
XXXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXXX

Exercice B -- 6 : options des lignes de commande Créer une fenêtre xterm de 30 caractères par 20 lignes.

Il suffit de lancer

% xterm -geometry 30x20

Placer une fenêtre "xeyes" dans le coin supérieur droit de l'écran à 30 pixels de chaque bord. La

fenêtre sera carrée de 150 pixels de côté.

Il suffit de lancer

% xeyes -geometry 150x150-30+30

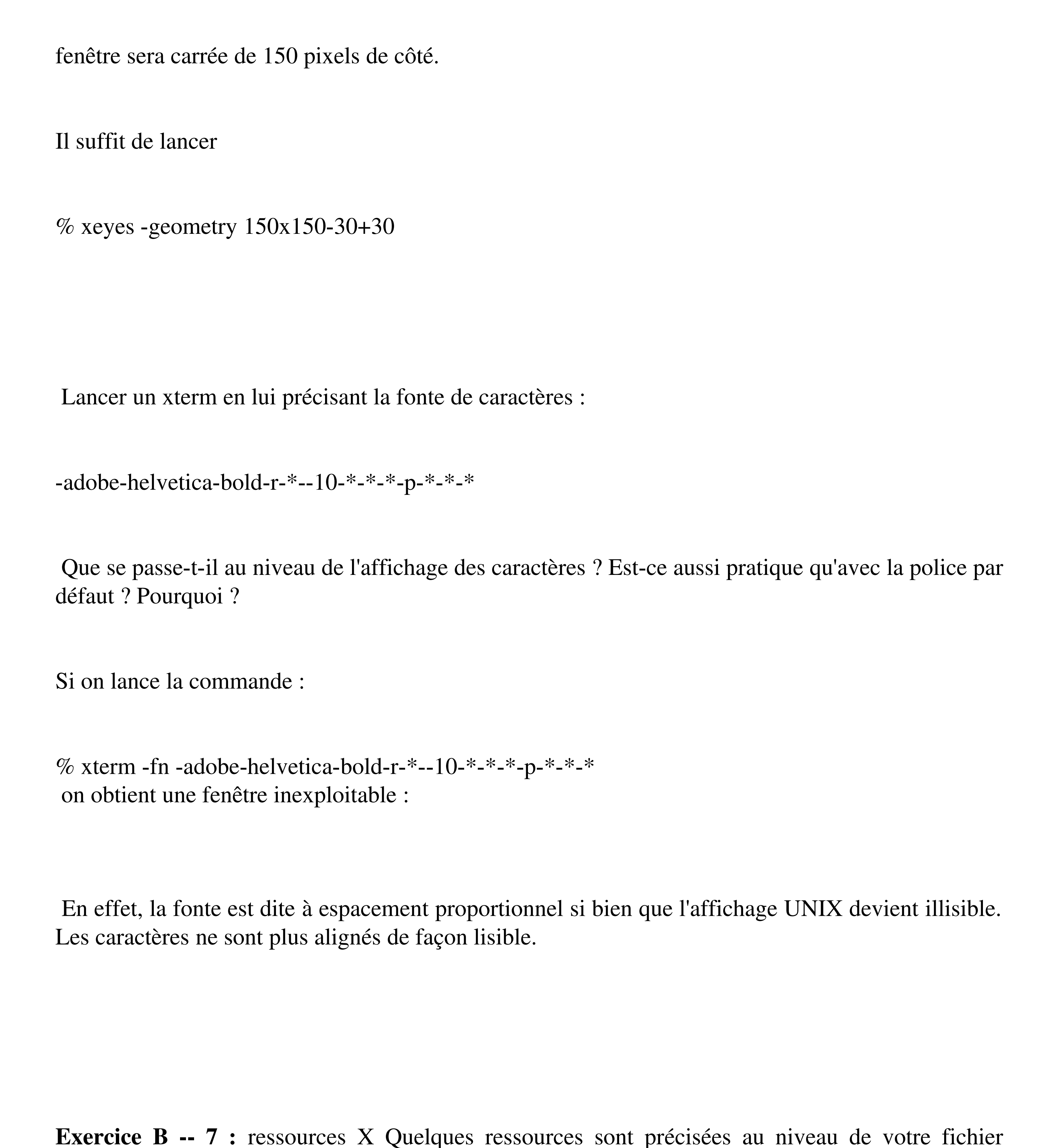
Lancer un xterm en lui précisant la fonte de caractères :

-adobe-helvetica-bold-r-*--10-*-*-*-p-*-*-*

Que se passe-t-il au niveau de l'affichage des caractères ? Est-ce aussi pratique qu'avec la police par défaut ? Pourquoi ?

Si on lance la commande :

% xterm -fn -adobe-helvetica-bold-r-*--10-*-*-*-p-*-*-*
on obtient une fenêtre inexploitable :



En effet, la fonte est dite à espacement proportionnel si bien que l'affichage UNIX devient illisible. Les caractères ne sont plus alignés de façon lisible.

Exercice B -- 7 : ressources X Quelques ressources sont précisées au niveau de votre fichier personnel ".Xresources".

Des ressources générales sont spécifiées au niveau de multiples fichiers dans "/usr/X11R6/lib/X11/app-defaults/" (sur Linux).

Changer la police de caractères par défaut de l'application xterm en modifiant votre fichier

".Xresources".

La modification est-elle active tout de suite ?

Si non, utiliser la commande "xrdb" pour indiquer au serveur X de stocker les ressources de votre fichier ".Xresources".

Au niveau de votre fichier ".Xresources", mettre les lignes :

```
XTerm*Font:              10x20
XTerm*BoldFont:          10x20
```

L'application xterm reconnaît l'option "-xrm". Utilisez la pour changer la police de caractères pour une instance de xterm.

Il suffit de lancer :

```
% xterm -xrm 'XTerm*Font: 10x20'
```

Séance 10 de travaux pratiques

Les devices UNIX

Il est conseillé de lire la totalité des sujets avant de commencer les exercices.
Avant propos

Il est conseillé de lire la totalité des questions d'un exercice avant de commencer l'exercice.

Créez le répertoire «tp9» et travaillez dedans afin de ne pas polluer votre homedir avec les fichiers générés par les exercices.

La plupart des opérations à réaliser doivent l'être sous l'identité de root puisque ce sont des commandes d'administration système.
Le mot de passe du compte root vous sera communiqué à l'occasion du TP. Merci de ne pas le divulguer.

Exercice A -- 1 : à propos de «/dev/null» Les machines UNIX fournissent une espèce de trou noir : «/dev/null»
Tout ce qui y est redirigé disparait à jamais. C'est comme un trou noir...

Constatez cette particularité avec des commandes du type :
echo oui > /dev/null
cat /etc/passwd > /dev/null
Que pensez-vous voir ?

N'importe qui doit pouvoir écrire dans «/dev/null». Quels sont les droits d'accès à «/dev/null» à votre avis ? Vérifiez les.

Recherchez chez l'utilisateur « sow», les fichiers textes nommés « toto ».
Indication : vous devez en trouver 3.

En dehors des 3 lignes indiquant les fichiers «toto» existants, quelles sont les lignes affichées et pourquoi ?

Ces lignes sont écrites sur le file descriptor 2, qui correspond au file descriptor «stderr».

Modifiez la commande de recherche précédente pour que désormais ces lignes d'erreur ne s'affichent plus
en les redirigeant vers «/dev/null».

Créez un fichier «toto» non vide, faites «ls -l toto» puis attendez plus de 60 secondes.
Faites alors «touch toto». Que constatez-vous comme différence au niveau de «ls -l toto» ?
Que fait la commande «touch» lorsque le fichier passé en paramètre existe déjà ?

Faites maintenant «touch truc». Que fait la commande «touch» lorsque le fichier passé en paramètre n'existe pas déjà ?

Que fait «cat /dev/null > truc2» ?
Que fait «cat /dev/null > toto» ? (avec le fichier «toto» non vide créé plus haut)
Déduisez en ce que fait «cat /dev/null > fichier» suivant que «fichier» existe ou pas ? Comparez ce comportement avec celui de la commande «touch».

Conventionnellement, les fichiers device sont dans «/dev». Tous les utilitaires sont programmés pour cela. Mais rien n'empêche de placer des fichiers device ailleurs. Pour les besoins du TP, recréez dans «/tmp» le device «/dev/null». Utilisez pour cela la commande «mknod». Appelez le nouveau device créé «blackhole». Est-ce qu'un simple utilisateur peut créer un device ? Vérifiez que le nouveau device peut être utilisé de la même façon que le «/dev/null» original (cf début de l'exercice).

Exercice A -- 2 : à propos de /dev/console et de /dev/tty Si vous travaillez en mode graphique (sous KDE ou sous GNOME), ouvrez une nouvelle session en mode texte. La méthode a déjà été vue à a la séance 9.

A qui appartient «/dev/console»

Que se passe-t-il si vous faites «cat /etc/passwd > /dev/console» ? (indice : la console est celle obtenue par «Alt» «F1»)
Connectez-vous en tant que root via «Alt» «F3».
Que se passe-t-il si vous faites «cat /etc/passwd > /dev/console» ?

Travaillez maintenant sous KDE ou GNOME.
Ouvrez un terminal. Qu'y renvoit la commande «tty» ?
Retrouvez-vous les noms de ces terminaux dans la commande «ps -ax» ?

Dans un terminal, lancez «tty». Votre terminal utilise-t-il un device en mode caractère ou en mode bloc ? Vérifiez-le.

Dans votre terminal, lancez «cat /etc/passwd». Que voyez-vous logiquement se passer ?
Dans votre terminal, lancez maintenant «cat /etc/passwd > /dev/tty». Que voyez-vous se passer ?
Que pensez-vous de ce device ?

Exercice A -- 3 : à propos de «/dev/zero» Les machines UNIX fournissent une espèce de générateur de bytes : «/dev/zero»

On peut y lire autant qu'octets qu'on veut. C'est un puits inépuisable d'octets prenant la valeur «\000».

Constatez la nature de ce fichier grâce à la commande :

dd if=/dev/zero bs=10k of=espace count=10

Expliquez la commande et ce qu'elle fait après avoir regardé sa page de manuel.

Vérifiez par les commandes suivantes que le contenu du fichier généré ne contient bien que des caractères de valeur «\000» :

od espace

od -a espace

Exercice A -- 4 : à propos des «sparse files» Soit le fichier C suivant («~sow/tp9/sparse.c») :

```
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <stdio.h>

main(int argc, char *argv[])
{
  int    fd;
  off_t  offset;
  char   *buf = "foo";
  if ( argc != 2 )
    exit(1);
  fd = open(argv[1], O_WRONLY | O_CREAT, 0644);
  offset = 16 * 1024 * 1024 ;
  lseek(fd, offset, SEEK_SET);
  write(fd, buf, strlen(buf));
  close(fd);
}
```

Que fait le programme à votre avis ?

Compilez le. Comment le lancer ? (regardez le code C pour comprendre)

Dupliquez le fichier sparse généré. On utilisera pour cela la commande Linux «cp --sparse=never file1 file2».

Constatez que le premier fichier est bien sparse alors que le second ne l'est plus. Utilisez pour cela l'option «-s» de «ls» . Essayez d'expliquer les nombres de ko réels renvoyés par « ls -ls » (question difficile).

Dupliquez à nouveau le fichier sparse généré. On utilisera pour cela la commande Linux «cp file1 file2».
Constatez que les deux fichiers sont bien sparse.

En déduire le comportement de la commande «cp» sur Linux.

Exercice A -- 5 : à propos de directories Un peu de programmation C : écrivez un programme C qui prend en paramètre un nom de répertoire et affiche les entrées du répertoire ainsi que leurs types. Point de départ : fonction «opendir()».

Comparez la sortie de votre programme avec la sortie de «find . -print» et la sortie de «ls -al». Que constatez-vous ? Avez-vous une idée expliquant la sortie non classée par ordre alphabétique de votre programme ?

Séance 11 de travaux pratiques

CRON

A propos des exercices sur crontab L'objectif est de vous faire manipuler crontab au cours du TP. Vous programmerez des actions pendant la matinée de façon à les voir se réaliser. Leurs heures de lancement dépendront de votre vitesse pour aborder cette série d'exercice. L'énoncé restera donc dans le vague sur le moment de lancer les actions. Vous vous adapterez.

Exercice A -- 1 Sur LINUX, le système cron laisse des traces dans le journal /var/log/cron. Vérifiez que le système est bien programmé de cette facon.

Exercice A -- 2 Rappelez-vous le mécanisme de fonctionnement de crontab. Au besoin, consultez la page de manuel de « crontab » sur votre machine LINUX. Déduisez-en comment on lance à certaines heures des actions.

Exercice A -- 3 Programmez l'affichage sur « /dev/console » toutes les minutes de l'heure. C'est vous en tant qu'utilisateur normal qui serez responsable de cette action. Vérifiez que le job est bien enregistré. Vérifiez son fonctionnement une fois programmé.

Lorsque vous travaillez sur l'environnement graphique des machines ARS, l'accès à « /dev/console » n'est pas forcément simple. Une solution simple est de lancer « xterm -C » qui a pour effet d'ouvrir une fenêtre terminal associée à « /dev/console » de votre machine.

Exercice A -- 4 Restreignez à root l'emploi de crontab. Au besoin, reportez-vous au cours pour savoir comment restreindre.
Est-ce que le job crontab utilisateur de l'exercice précédent continue de tourner ? Pour cela, regardez par exemple le fichier « /var/log/cron ».

Séance 12 de travaux pratiques

Les filesystems UNIX

Avant-propos : L'objet de ce TP est de réaliser sur disquette des opérations que l'on réaliserait en principe sur un disque dur avec un système installé dessus.

Dans la mesure où ces opérations nécessiteraient un disque additionnel que nous n'avons pas ou dans la mesure où ces opérations pourraient endommager le disque dur, il est préférable de les réaliser sur disquette, tout en sachant que les résultats peuvent se généraliser.

Exercice A -- 1 : formattage de disquette Sur LINUX, effectuez un formatage de bas niveau sur une disquette PC. La commande à employer est "fdformat". Quel est le nom de périphérique désignant le lecteur de disquettes ? (cherchez dans la page de manuel de «fdformat»)

Exercice A -- 2 : mkfs, ext2, mount, fstab, df, umount, /mnt Sur une machine LINUX, déposez un filesystem UNIX sur la disquette. Pour cela, reportez-vous à la commande "mkfs". Le filesystem à déposer sur la disquette sera de type "ext2" (le type UNIX par défaut sur LINUX).

ATTENTION pour la question qui suit : le système graphique dans son désir de simplifier la vie de l'utilisateur, permet de monter une disquette en cliquant simplement sur l'icone KDE du lecteur. A priori dans cet exercice, il s'agira plutôt de lancer les commandes à la main.

A priori, il n'est pas néssaire de quitter le runlevel 5 pour revenir au runlevel 3 sans interface graphique. Il n'y a pas de réelle gêne en pratique par le mécanisme de l'icone montable.

Montez le filesystem nouvellement créé. Employez pour cela la commande "mount" sous sa forme précisant le device et le point de montage. On fera le montage sur "/mnt/floppy".
Démontez la disquette puis procédez à nouveau au montage en employant la commande "mount" sous sa forme utilisant le fichier "/etc/fstab". Finir par démonter la disquette.

La disquette doit être maintenant démontée. Montez à nouveau la disquette mais au moyen de l'interface graphique (si vous l'avez tuée via « init 3 », repassez dans le runlevel 5). Démontez la disquette après avoir consulté son contenu.

Remontez à nouveau la disquette de la façon dont vous voulez. Affichez le nombre d'inodes utilisé. Pour cela, reportez vous à la commande "df". Que contient le filesystem ? Est-ce que la taille des fichiers déjà présents vous semble normale ? A quoi servent-ils ? Démontez la disquette.

Créez l'arborescence «/mnt/test/a».
Montez le filesystem de la disquette créé précédemment sous «/mnt/test» (attention au montage automatique de la disquette via le système graphique).
Voyez-vous encore «/mnt/test/a» ?
Démontez le filesystem. Voyez-vous encore «/mnt/test/a» ?

Effacez l'arborescence «/mnt/test/».

Exercice A -- 3 : sync, umount Copiez un fichier de quelques centaines de ko sur le filesystem monté de la disquette. Par exemple le fichier "/boot/vmlinuz" ou 2 copies concaténées pour avoir un fichier de plus gros.

Pourquoi les diodes d'accès à la disquette s'allument-elles encore après la fin de la commande UNIX ?

Refaites la commande de copie d'un fichier.
Lancez immédiatement après, la commande "sync". Que se passe-t-il ?

Refaites la commande de copie d'un fichier.
Démontez le filesystem juste après. Que se passe-t-il ?

Exercice A -- 4 : droits d'accès, /dev Cet exercice commence par réinitialiser la disquette à zéro. Pour cela, refaites le "fdformat" puis le "mkfs".

Montez la disquette quelque part dans l'arborescence, comme précédemment.
Copiez ensuite un fichier sur la disquette, le fichier "/etc/passwd" par exemple ou un de vos fichiers C (en n'importe quel fichier de type texte du moment que vous pourrez en reconnaitre le contenu).
Démontez la disquette.

Faites maintenant "more /dev/fd0". Que finissez-vous par reconnaître par cette commande ?

Qu'en déduisez-vous sur les droits d'accès que l'on doit donner en général aux fichiers spéciaux de "/dev" associés aux périphériques de type disque/disquette ?
D'ailleurs quels sont les droits du périphérique lecteur de disquette ?

Exercice A -- 5 : /dev, mknod, /dev/MAKEDEV Conventionnellement, les fichiers devices se trouvent dans le répertoire "/dev". C'est purement conventionnel.

Créez le répertoire "/tmp/dev".
Dedans, recréez les devices associés aux lecteur de disquettes 0. Vous procéderez d'abord manuellement, en utilisant la commande "mknod" puis de façon automatique via le script "/dev/MAKEDEV" que vous aurez recopié dans "/tmp/dev" (lisez la page de manuel de "/dev/MAKEDEV").

Vérifiez que l'on peut manipuler la disquette en utilisant les fichiers spéciaux que vous avez crées.

Exercice A -- 6 : dd Cet exercice commence par réinitialiser la disquette à zéro. Pour cela, refaites le "fdformat" puis le "mkfs".

Montez la disquette quelque part dans l'arborescence, comme précédemment.
Copiez ensuite le fichier "/etc/passwd" sur la disquette.
Démontez la disquette.

Faites maintenant "dd if=/dev/fd0 of=/tmp/image bs=128k". Que pensez-vous avoir obtenu ?

Réinitialisez à nouveau la disquette à zéro et déposez dessus un filesystem. Vérifier que la disquette ne contient rien comme fichier.

Faites la commande "dd if=/tmp/image of=/dev/fd0 bs=128k". Que pensez-vous avoir obtenu ?

Cet exercice vous montre la façon dont on duplique une disquette sur UNIX. On utilise couramment cette opération lors de l'installation LINUX pour créer les disquettes de boot.

Exercice A -- 7 : fsck (utiliser une disquette qui ne risque rien)

On va chercher dans cet exercice à créer une disquette contenant un filesystem avec des incohérences puis voir ce que la vérification de cohérence donne. On retirera pour cela la disquette alors que certaines opérations du filesystem n'auront pas encore été faites de façon à obtenir une incohérence. Cette façon de faire ne marche qu'avec des disquettes mais simule assez ce qui pourrait arriver pour d'autres raisons avec un disque dur.

Partez d'une disquette avec un filesystem UNIX vierge. Refaites pour cela le "fdformat" puis le "mkfs".

Copiez un gros fichier (de taille supérieure à 1 Mo, au besoin créer le fichier) sur la disquette et retirez la disquette du lecteur avant que la copie ne soit terminée.
La disquette toujours retirée, faites "sync" puis lancez la commande "df".

Essayez de démonter le filesystem (la disquette étant encore absente du lecteur).

Une fois que la situation est redevenue normale (il y aura certainement des messages d'erreurs à l'écran), remontez la disquette (après l'avoir remise dans le lecteur).

Pouvez-vous monter la disquette ?
Son contenu vous parait-il normal ?
Vérifiez les "df" avant et après avoir effacé le fichier créé. Est-ce que la disquette vous parait effectivement vide après l'effacement ?
Si vous voyez une anomalie au niveau du « df » après l'effacement, utilisez la commande "e2fsck" pour corriger un problème de cohérence s'il y en a.

Séance 13 de travaux pratiques

SWAP

Exercice A -- 1 : dd, /dev/zero Sur Linux, créez un fichier de 16 Mo (ou approchés) dans « /tmp ». Utilisez pour cela la commande « dd » avec pour « if » la source « /dev/zero ». On utilisera les options « bs » et « count ».

Ajoutez ces 16 Mo à l'espace de swap. Pour cela, utilisez la commande « mkswap » puis « swapon ». On vérifiera que le mécanisme de swap a bien hérité des 16 Mo. Pour cela, réfléchissez sur comment constater que la taille du swap a augmenté.

Supprimez ces 16 Mo ajoutés sans rebooter la machine. Quelle commande à votre avis faut-il utiliser ?

Exercice A -- 2 : swapon, script de démarrage Trouvez dans quel script de démarrage sur Linux la commande « swapon » est lancée au démarrage.

Exercice A -- 3 : vmstat, ps, top, free, meminfo Testez les commandes suivantes sous 1 permettant de s'informer sur la mémoire virtuelle de la machine et son état à un instant donné :

/usr/bin/vmstat
/bin/ps
/usr/bin/top
/usr/bin/free
cat /proc/meminfo

Exercice A -- 4 : C, alloc() Cet exercice a pour but de créer un programme C sur UNIX dont le but est d'allouer de la mémoire virtuelle pour voir comment réagit le système de swap.

Ce programme servira à l'exercice suivant. Afin de ne pas être bloqué, je fournis un programme fonctionnel qui permet de faire la suite de ce TP mais, dans un premier temps, essayez d'écrire ce programme vous même.

Le programme demandé doit répondre aux points suivants :

Le programme doit prendre 2 paramêtres depuis la ligne de commande.
Un premier paramêtre passé depuis la ligne de commande sera un nombre N de mega octets à allouer.
Le second paramêtre passé depuis la ligne de commande sera une durée en secondes pendant laquelle on gardera alloué l'espace de N mega octets.

Après cette durée, on libérera la mémoire préalablement allouée.
Le programme affichera des indications sur l'état d'avancement des choses.

Rappel :

Un programme C sous UNIX a un nom qui se termine par le suffixe « .c ».
Le compilateur C sous UNIX s'appelle pour un programme simple par :
gcc toto.c -o toto

Les fonctions C sous UNIX sont documentées comme n'importe quelle autre commande UNIX. On obtient leur documentation via la commande classique « man ».
En pratique il faut initialiser la mémoire allouée pour l'allouer pour de bon et pour voir que le système consomme du swap. On utilisera pour cela la fonction «memset() » qui remplit une zone mémoire avec un caractère passée en paramètre.
La fonction C pour attendre N secondes est «sleep() ».

Mon programme C manipulant de la mémoire virtuelle s'appelle « tp/11/hog.c », chez l'utilisateur « sow ».

Au besoin, recopiez le chez vous et poursuivez le reste de cet exercice avec.

Au moyen de ce programme C, allouez de la mémoire virtuelle dans divers cas :

allouez de la mémoire virtuelle tenant en mémoire vive
allouez de la mémoire virtuelle tenant en mémoire vive de justesse
allouez de la mémoire virtuelle ne tenant pas en totalité en mémoire vive
allouez de la mémoire virtuelle que le système de swap ne pourra pas attribuer en pratique

Séance 14 de travaux pratiques

DUMP / RESTORE / TAR

Exercice B -- 1 : fdformat, mkfs, mount Travaillez en tant qu'administrateur.

Partir d'une disquette neuve, la formater, déposer dessus un filesystem de type "ext2", monter les filesystème, y créer quelques fichiers, ne pas démonter la disquette.

Si vous constatez que les commandes dump et restore ne sont pas disponibles, réinstallez les. Pour cela suivez ces instructions :

travailler en tant que root
travailler dans /tmp
récupérer le fichier dump-0.4b27-3.i386.rpm en cliquant ici
installer le logiciel via la commande spécifique Red Hat LINUX :
rpm -ivh dump-0.4b27-3.i386.rpm ou bien :

travailler en tant que root
installer le logiciel via la commande spécifique Red Hat LINUX :
apt-get install dump

Exercice B -- 2 : dump, restore Travaillez en tant qu'administrateur.

Réalisez une sauvegarde de la disquette au moyen de la commande "dump" sur Linux. On mettra le fichier de la sauvegarde dans le répertoire "/tmp".

Dans un répertoire de votre choix, récupérer dans la sauvegarde l'un des fichiers que vous avez crées sur la disquette. On employera l'option "-i" de "restore" pour faire cela.

Exercice B -- 3 : tar Travaillez en tant qu'administrateur.

Réalisez une sauvegarde de la disquette au moyen de la commande "tar" sur Linux. On mettra le fichier de la sauvegarde dans le répertoire "/tmp".

Exercice B -- 4 : GNU TAR (1)Travaillez en tant que simple utilisateur.

Sur LINUX, l'utilitaire tar par défaut correspond en fait au logiciel GNU TAR.

Lisez la documentation du GNU TAR. Pour cela, lancez la commande « tar --help » (on notera au passage que bon nombre de commandes GNU utilisent cette option « --help » pour afficher des informations). Dans la documentation, lisez en particulier le passage sur les options de compression.

Exercice B -- 5 : GNU TAR (2) Travaillez en tant que simple utilisateur.

Au moyen de l'utilitaire « tar », réalisez une archive de l'arborescence « /etc/ ». Vous mettrez cette archive dans le répertoire « /tmp ». Compressez ensuite cette archive au moyen de GNU ZIP.
sauvegarde. Le message final résume cela.

Reprenez la question précédente en réalisant en une seule opération l'archivage et la compression.

Exercice B -- 6 : GNU TAR (3) Travaillez en tant que simple utilisateur.

Récupérez le logiciel disponible via l'adresse « ftp://ftp.irisa.fr/pub/gnuplot/libpng-1.0.11.tar.gz ». Désarchivez le après avoir pris la précaution de regarder sous quels noms seront créés les fichiers désarchivés (afin de ne pas trop polluer les fichiers existants).

Exercice B -- 7 : GNU TAR (4) Travaillez en tant que simple utilisateur.

Récupérez le logiciel disponible via l'adresse
« http://www.php.net/distributions/manual/php_manual_en.tar.bz2 ». Désarchivez le après avoir pris la précaution de regarder sous quels noms seront créés les fichiers désarchivés (afin de ne pas trop polluer les fichiers existants).

Séance 15 de travaux pratiques

Les comptes utilisateurs

Exercice A -- 1 : LINUX, shadow passwords Le système LINUX des machines de la salle TP utilise-t-il le mécanisme des shadow passwords ? Si oui, quels fichiers gérent-ils finalement les comptes utilisateurs ?

Exercice A -- 2 : création manuelle d'un compte L'objectif de cet exercice est de créer un compte utilisateur manuellement sans passer pour l'instant par des commandes dédiées spéciales à LINUX. Ce compte ne sera pas rentré dans le système NIS, ce sera un compte local à chaque machine de la salle de TP ARS.

En vous aidant au besoin du cours, retrouvez les différentes étapes en lesquelles consistent une création de compte.

Après, créez un compte dénommé « localuser » dans le groupe « localgroup » qui aura un homedirectory dans « /tmp » (on passera sur les fichiers d'initialisation « .machin »).

Vérifiez que le compte ainsi créé fonctionne correctement (login possible, connexion graphique opérationnelle).

Effacez le compte créé une fois cet exercice terminé.

Exercice A -- 3 : création assistée d'un compte Créez à nouveau un compte utilisateur en utilisant cette fois-ci la commande système LINUX «useradd ».

Détruisez le compte utilisateur dernièrement crée en utilisant la commande système LINUX «userdel ».

Utilisez les commandes système graphiques pour créer un compte puis l'effacer.

Exercice A -- 4 : LINUX, root, observez dans le fichier local « /etc/passwd » s'il y a des comptes intitulés « root » et « toor ». Qu'ont-ils comme caractéristiques ? Fonctionnent-ils ? Pourquoi à votre avis a-t-on ces 2 comptes ?

Exercice A -- 5 : tri de /etc/passwd Triez à l'écran les comptes UNIX par UID croissant, puis par UID décroissant, puis par login croissant.

Exercice A -- 6 : join, Lisez la page de manuel de la commande UNIX « join ». lisez notamment soigneusement les contraintes sur les champs de jointure.

En vous appuyant sur cette commande, générez les données des comptes sous le format sensible des UNIX primitifs : le format « <pwd.h> » qui contient la chaine hashée du mot de passe. Sous quelle identité faire cela ? Pourquoi ? Plusieurs commandes UNIX étant nécessaires pour arriver au résultat demandé, faites un shell script pour résoudre l'exercice.

Exercice A -- 7 : chown, chgrp Travaillez en tant qu'utilisateur de base.

Copiez le fichier ~sow/tp15/show-uid.c chez vous. Quels sont l'UID et le GID du fichier maintenant chez vous ?

Au moyen des commandes « chown » et « chgrp », pouvez-vous passer au nom de l'utilisateur « sow » et du group « wheel » la copie chez vous du fichier ?

Refaites en tant qu'utilisateur de base la copie du fichier précédent mais en le copiant dans « /tmp ». Réessayez après les commandes « chown » et « chgrp » en tant que root sur la copie dans « /tmp ». Est-ce OK ?

Travaillez maintenant à nouveau sur le fichier copié chez vous dans « ~/tp15 ». Réessayez les commandes « chown » et « chgrp » en tant que root sur votre copie c'est-à-dire sur ce fichier résidant sur un disque réseau. Est-ce OK ? (l'explication de ce qui se passe sera vu au tome 4 du cours UNIX, ne cherchez donc pas trop longtemps, contentez vous d'émettre une hypothèse basée sur ce les faits)

Faites le ménage dans « /tmp » une fois cet exercice terminé.

Exercice A -- 8 : UID inexistant, find Sur votre machine de TP, créez dans « /tmp » un fichier. Faites ensuite les commandes suivantes en étant root :

chown 3333 /tmp/foo
chgrp 4444 /tmp/foo
Utilisez en fait un UID et un GID qui ne sont pas en service. Pour en être sûr, reportez-vous au fichier « /etc/passwd » et à la base de données NIS (obtenue par « ypcat passwd »).

Que vous renvoit un « ls -l /tmp/foo » ?

Recherchez avec la commande « find » dans « /tmp » les fichiers qui appartiennent à un UID pour lesquels il n'y a pas d'utilisateur associé. Pour cela, reportez vous à la documentation de « find » et cherchez y l'option qui convient.

Faites le ménage dans « /tmp » une fois cet exercice terminé.

Exercice A -- 9 : mots de passe, su, root Travaillez en tant que root sur votre machine (hypothèse de l'énoncé).

Pouvez-vous prendre l'identité d'un autre utilisateur ? Quelle commande utiliseriez-vous pour faire cela ?

Comprenez-vous maintenant pourquoi la compromission du compte root est un tel objectif pour les pirates ?

Exercice A -- 10 : su, su - Travaillez en tant qu'utilisateur de base. Passez sous l'identité de root en faisant « su ». Réussissez-vous à lancer la commande « ifconfig -a » ?

Travaillez en tant qu'utilisateur de base. Passez sous l'identité de root en faisant « su - ». Réussissez-vous à lancer la commande « ifconfig -a » ?

Expliquez ce qui se passe dans les deux cas. Quelle moralité en extraire ?

Exercice A -- 11 : su et syslog En étant connecté sur une machine de la salle de TP ARS sous une identité normale, exécutez un « su root » ou un changement d'identité vers un autre compte d'ARS.

Analysez ce qui se passe alors au niveau du « /var/log/messages ».

Exercice A -- 12 : groups, id, whoami Que font les commandes « groups », « id », « whoami », « who am i » ?